Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Gruppe Softwaretechnologie
http://www-st.inf.tu-dresden.de
Content

► Introduction
  ▪ What is software engineering?
► Analysis and Validation
  ▪ Requirements Analysis
  ▪ Testing
  ▪ Inspections
  ▪ Cost Estimation
► Development Methods
  ▪ Standard design
  ▪ Advanced object oriented design
  ▪ Component-based design
► Software Evolution
  ▪ Process Models
  ▪ Maintenance
► Earning Money with Software
► Saving Costs with Software Reuse
  ▪ Configuration Management
  ▪ Product lines
  ▪ Model-driven architecture
Recommended Literature:
Overview Books

► We recommend one of (reading instructions can be followed in one of them):

► Other good books, priority from top to bottom:
  ▪ I. Summerville, Software Engineering. Addison-Wesley, 7th edition
Recommended Books on UML – Unified Modelling Language

We recommend one of:
- Online documentation on [www.omg.org/uml](http://www.omg.org/uml)

Other excellent books:
- Ken Lunn. Software development with UML. Palgrave-Macmillan. Many case realistic studies.
- Mathiassen et. al. Objektorienterad analys och design. Studentliteratur. Pretty good introduction into UML.

UML is required. It is expected that you learn UML yourself from a good book.
Reference Books


Analysis and Design

This standard reference book belongs to the bookshelf of every software engineer!
Buy this now, if you want to visit TDDB84. To 50%, TDDB84 is build on it.

http://www.omg.org.uml Or any other good book on UML. There are so many, choose one of them.

Others
Rumbaugh et.al. Object-oriented modelling and design. Prentice-Hall.
In German: Heide Balzert. Objektorientierten Systemanalyse. Spektrum der Wissenschaft.

Component-Based Design

► C. Szyperski: Component Software. Addison-Wesley
► K. Czarnecki, U. Eisenecker: Generative Programming. Addison-Wesley
Project Management

► B. W. Boehm, Software Risk Management, 1989
► Brooks, The Mythical Man-Month, Addison-Wesley, 1975
► E. Yourdan: The Death March.
► David Thielen. The 12 simple secrets of Microsoft McGraw-Hill.
Implementation, Metrics

Testing, Quality

Configuration Management

▶ Version control with subversion. http://svnbook.red-bean.com/, also available as paper book of O'Reilly


Miscellaneous Literature

On Writing:
Fogler/LeBlanc, Strategies for Creative Problem Solving
Warning:
Remarks on the Nature of the Course

A University is unlike a high school
   You should not expect to get a book, and that's it
Software Engineering is too broad for that, unfortunately
   The lectures have to focus on most important things
   You should not expect to be an expert after the course
Find your way from the lecture slides into the book
   Follow the reading instructions
Learn the additional material and read the additional readings
Follow the exercise sheets
   Solutions will be explained in the storseminarier

➤ **Be aware: you have not yet seen larger systems**
   - Middle-size systems start over 100KLOC
Remarks on the Nature of the Course

The purpose of lecturing is
To give you a condensed insight on the most important topics, such that you do not waste too much time during reading
To give you pointers for future work, once you left the course
   If you haven't got the pointer, you can waste years in darkness
To give you experience from practice of the lecturer (you can't read in books)
   These stories are sometimes very important for critical situations in your professional career
To be honest, stories is sometimes the only thing you remember
Softwaretechnologie II
Goals

Prof. Dr. U. Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Gruppe Softwaretechnologie
http://www-st.inf.tu-dresden.de
Main Goals

► Learn about “engineering” software
  ▪ Engineering attitudes
  ▪ Technology, process, experiences, human conditions
► Get as many ideas as possible (broad overview)
  ▪ NOT: technical in-depth teaching (this must be left to other courses)
► Get an introduction into the main obstacle: from a set of requirements, how do I arrive at a system?
  ▪ (forward engineering)
► Teach you systematic methods for simple, graph-based specifications
  ▪ Because almost all requirements and design notations are graph-based
  ▪ Get hold on the complexity of a large specification
Design

Know different forms of design methods
  functional, object-oriented, data-oriented
Know about “software architecture” and architectural styles
Know behavioral methods to generate code for verifiable specifications
  Petri nets
  UML profiles
Processes

Get simple overview of software processes
MDA, XP, V-model,
EOS, ....
Know what inspections are
Know about maintenance problems
Know about requirement specification
Know about basic testing concepts
Other Courses

Design Patterns and Frameworks (WS)
- Basic design patterns
- Design patterns in frameworks
- Role-based design
- Composition of design patterns
- Layered frameworks

Component-based Software Engineering (SS)
- Black-box component models (e.g., EJB)
- Grey-box component models (e.g., aspects)
- Software composition
The End